Technical Supplement ~ 7

RESTARTABILITY REVISITED

METHODS OF WRITING 'RESTARTABLE' FUNCTIONS WERE DISCUSSED IN THE MARCH/APRIL 76 ISSUE OF THE SUPPLEMENT.
THE IDEA WAS TO RESTART AT THE 'RIGHT' POINT AFTER A SYSTEM CRASH. WITH THE ADDITION OF A FUNCTION RETYING ANY NEEDED FILES IT WAS HOPED THAT THE WS COULD BE MADE RESTARTABLE.

IN THIS ISSUE WE WILL DISCUSS SITUATIONS WHERE APL CODING CAN JEOPARDIZE RESTARTABILITY.

A SYSTEM CRASH LOSES THE 'CURRENT' COPY OF ACTIVE WORKSPACES THAT WERE IN MAIN MEMORY. THE 'WS CRASH RECOVERY' PROCEDURE RETRIEVES THE MOST RECENT COPY OF EACH ACTIVE WORKSPACE FROM SECONDARY MEMORY (DRUM OR DISK). THIS COPY WAS WRITTEN TO SECONDARY MEMORY WITHIN 4 SECONDS OF THE CRASH AND MAY BE IN THE MIDDLE OF EXECUTING A LINE. AS A WORKSPACE CANNOT BE SAVED INTO 'CONTINUE' IN SUCH A STATE, THE WS IS PUT INTO EXECUTION TO TRY TO ALLOW FINISHING THE LINE SO THAT THE WS CAN BE SAVED READY TO EXECUTE THE NEXT EXECUTABLE STATEMENT.

THUS, PRIOR TO SAVING YOUR WORKSPACE IN 'CONTINUE',
'WS CRASH RECOVERY' EXECUTES THE MOST CURRENT LINE OF THE WS FROM THE POINT OF INTERRUPTION, WITH A SIMULATED LINE DROP AND BOUNCE WITHIN THE FOLLOWING CONSTRAINTS:

1) A MAXIMUM OF 60 CPU UNITS ARE ALLOWED BEFORE AN INTERRUPT IS CAUSED,

2) AS THE FILE SYSTEM IS NOT AVAILABLE, FILE OPERATIONS RESULT IN AN ERROR, WITH EXECUTION OF THE LINE INTERRUPTED,

3) ⎕ OR ⍞ ARE SET UP IN SI AND EXECUTION IS SUSPENDED.

THE WORKSPACE IS THEN SAVED INTO 'CONTINUE' WITH THE 'RECOVERED' MESSAGE AND TIME STAMP.

IF THE LINE EXECUTES SUCCESSFULLY, THE)SI IN THE 'CONTINUE' WS WILL POINT TO THE NEXT STATEMENT TO BE EXECUTED.
OTHERWISE THE LINE NUMBER IN THE)SI REMAINS UNCHANGED WITH THE EXCEPTION OF (3), WHERE)SI WILL POINT TO ⎕ OR ⍞, RESPECTIVELY.

AN EXAMPLE TO CLARIFY MATTERS:

```
    ∇ NICE
.
[2]   I←1 ◇ LIM←28
[3]   ⎕HOLD F
[4]  L1:A←⎕READ F,I
```

```
    ∇ UGH
[2]   I←1 ◇ LIM←28 ◇ ⎕HOLD F
[3]  L1:A←⎕READ F,I
```

ASSUME THAT BOTH FUNCTIONS WERE SWAPPED OUT ON LINE 2 AFTER THE ASSIGNMENT I←1.

DURING WS CRASH RECOVERY, THE FUNCTION 'NICE' WILL RESUME EXECUTION OF LINE 2 FROM THE POINT OF INTERRUPTION (I.E. AFTER I←1), AND COMPLETE IT SUCCESSFULLY, WITH)SI POINTING TO LINE 3 IN THE CONTINUE WS.

CONVERSELY, IN THE FUNCTION 'UGH', EXECUTION WILL TERMINATE UNSUCCESSFULLY IN LINE 2 AS A RESULT OF THE FILE COMMAND, WITH THE)SI POINTING TO LINE 2 IN THE CONTINUE WS.

APART FROM REDOING THE SPECIFICATION OF 'I' AND 'LIM' IN LINE 2 IN THE FUNCTION 'UGH', BOTH FUNCTIONS WILL WORK AS EXPECTED UPON →RESTART.

NOW, CONSIDER THE FOLLOWING EXAMPLE:

```
    ∇ NICE
.
[7]   →(LIM<I←I+1)ρEND
[8]   A←⎕READ F,I
```

```
    ∇ UGH
[7]   →(LIM<I←I+1)ρEND ◇ A←⎕READ F,I
```

ASSUME THAT BOTH FUNCTIONS WERE SWAPPED OUT IN LINE 7 AFTER I←I+1. AT WS CRASH RECOVERY, THE FUNCTION 'NICE' WILL SUCCESSFULLY CONTINUE EXECUTING LINE 7 FROM THE POINT OF INTERRUPTION AND HAVE)SI POINT TO LINE 8 IN THE CONTINUE WS.

ON THE OTHER HAND, UPON RESUMPTION OF EXECUTION, OUR POOR FUNCTION 'UGH' WILL INTERRUPT AS A RESULT OF THE FILE READ, WITH THE)SI POINTING TO LINE 7. UPON RESTARTING THE WS FROM CONTINUE, →RESTART WILL INCREMENT 'I' AGAIN, MAKING US WONDER WHY ONE READ WAS SKIPPED!

HERE ARE MORE EXAMPLES WITH UNEXPECTED MESSAGES THAT MIGHT APPEAR UPON →RESTART.

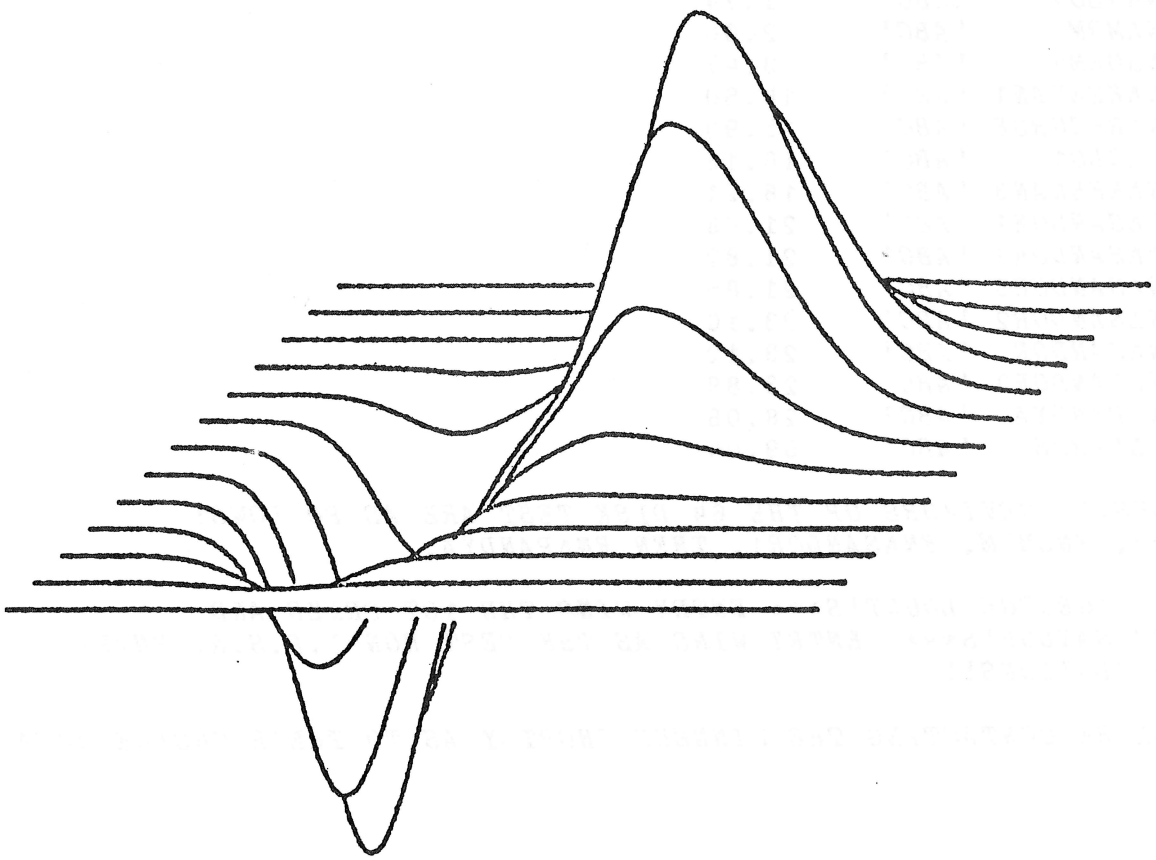
```
[N]   A←⎕READ X←X[1;1]                    'RANK ERROR'

[P]   A←⎕READ F,I←I+1 ◇ →(LIM<I)↑END      'FILE INDEX ERROR'
```

I WOULD BE INTERESTED IN OBTAINING OTHER EXAMPLES FROM USERS THAT LEARNED RESTARTABILITY THE HARD WAY!

SUMMARY: WS CRASH RECOVERY TRIES TO CONTINUE EXECUTION OF THE THE CURRENT LINE OF THE WS (WITH THE CONSTRAINTS STATED ABOVE) PRIOR TO PUTTING IT INTO 'CONTINUE'.

BE VERY CAREFUL WHEN INTERMIXING FILE OPERATIONS, IRREVERSIBLE ASSIGNMENTS, LAMINATIONS, ETC. ON THE SAME LINE.

FINALLY, FILES CAN BE AS MUCH AS 4 SECONDS 'AHEAD' OF A RECOVERED WS; THE NEXT ISSUE WILL DISCUSS THIS PROBLEM.



*** TOWER-OF-HANOI CONTEST RESULTS ***

WE WISH TO THANK ALL WHO PARTICIPATED IN THE CONTEST, AND INVITE YOU WHO CONSIDERED, BUT DIDN'T, TO TRY YOUR HAND AT THE NEXT ONE!

THE HANOI PARTICIPANTS AND THEIR ENTRIES WERE AS FOLLOWS:

ROGER HUI, I.P.S.A, CALGARY	FN∆HUI
GEORGE LOUNT, I.P.S.A, TORONTO	FN∆GLO1, FN∆GLO2
MARVIN MANDELBAUM, I.P.S.A., ROCHESTER	FN∆MRM - FN∆MRM2
DOUG KEENAN, UNIVERSITY OF WATERLOO	FN∆KEENAN1 - FN∆KEENAN3
MARC SANDOR, DEPT. OF PUBLIC WORKS, OTTAWA	FN∆SANDOR - FN∆SANDOR5
GAUTAM PANDYA, I.P.S.A., ROCHESTER	FN∆PANDYA
M.J. LAROS, AMSTERDAM, THE NETHERLANDS	FN∆LAROS

ALL FUNCTIONS WERE FIRST RANKED FOR SPEED OF EXECUTION WITH 7 DISCS, AND THEN CHECKED THAT THEY COULD HANDLE 64.

THE RESULTS OF THE RANKING ARE SHOWN BELOW, WITH THE EXECUTION TIMES EXPRESSED AS A MULTIPLE OF THE FASTEST ENTRY.

	SPEED 7		
7	FN∆HUI	'ABC'	1.00
7	FN∆KEENAN2	'ABC'	1.24
7	FN∆MRM2	'ABC'	1.74
7	FN∆GLO2	'ABC'	1.74
7	FN∆MRM	'ABC'	2.00
7	FN∆MRM1	'ABC'	3.47
7	FN∆KEENAN1	'ABC'	14.60
7	FN∆RECURSE	'ABC'	15.90
7	FN∆GLO1	'ABC'	16.16
7	FN∆KEENAN3	'ABC'	18.21
7	FN∆SANDOR1	'ABC'	21.08
7	FN∆SANDOR4	'ABC'	21.82
7	FN∆SANDOR5	'ABC'	21.86
7	FN∆SANDOR2	'ABC'	23.10
7	FN∆SANDOR	'ABC'	23.12
7	FN∆SANDOR3	'ABC'	23.89
7	FN∆PANDYA	'ABC'	28.06
7	FN∆LAROS	'ABC'	69.04

THE 'BEST' SURVIVORS OF THE 64 DISK TEST ARE AS FOLLOWS:
FN∆GLO2, FN∆MRM, FN∆SANDOR1, THEN FN∆PANDYA.

THUS, ***GEORGE LOUNT'S*** ENTRY WINS THE 1ST PRIZE AND
MARC SANDOR'S ENTRY WINS AS THE BEST NON-I.P.S.A. ENTRY.
CONGRATULATIONS!!

WE WILL BE CONTACTING THE WINNERS SHORTLY AS TO THEIR CHOICE OF A BOOK PRIZE.

ALL ENTRIES CAN BE FOUND IN THE WORKSPACE 1860450 HANOI.
*THE TWO WINNING FUNCTIONS, AND ALSO ***ROGER HUI'S*** VERY ELEGANT ENTRY, ARE LISTED BELOW.*

THANKS AGAIN FOR PARTICIPATING!

```
      ∇FN∆HUI[□]∇
    ∇ N FN∆HUI Z;DISK;I;PEG
[1]   I←DISK←1 ◇ PEG← 1 2 ρ 1 3
[2]   →TEST
[3]  LOOP:DISK←DISK,I,DISK
[4]   PEG← 1 3 2[PEG],[1] 1 3 ,[1] 2 1 3[PEG]
[5]  TEST:→LOOP×⍳N≥I←I+1
[6]   'G□MOVE DISK Z9□,G□ FROM 9□,G□ TO 9□' □FMT(DISK;PEG)
    ∇

      ∇FN∆GLO2[□]∇
    ∇ N FN∆GLO2 C;□IO;B;BL;D;F;FV;I;J;K;M;P;PV;S;S1;T;V;W
[1]   →(N≤0)↑□IO←0 ◇ S1←¯1+S←2*M←N⌊¯6+⌈2⊗□WA ◇ T←2*N
[2]   FV←Nρ 1 ¯1 ◇ PV←×\B←Nρ2 ◇ D←F←P←⍳0 ◇ I←1
[3]  L1:D←D,I,D ◇ F←F,FV[I-1],F ◇ P←P,PV[I-1],P ◇ →(M≥I←I+1)ρL1
[4]   D←D,0 ◇ F←F,0 ◇ P←P,0 ◇ I←1↓⍳S+1
[5]  L2:→(BL←T=I[S1])↓L3 ◇ I←S1↑I ◇ D←S1↑D ◇ P←S1↑P ◇ F←S1↑F ◇ →L4
[6]  L3:D[S1]←1+J←(⍬⍴T⍴I[S1])⍳1 ◇ F[S1]←FV[J] ◇◇ P[S1]←PV[J]
[7]  L4:V←3|F×W←⌊I÷P ◇ W←3|F×W+1
[8]   '□ MOVE DISK □,I1,□ FROM □,A1,□ TO □,A1' □FMT(D;C[V];C[W])
[9]   →BL↑0 ◇ I←I+S ◇ →L2
    ∇

      ∇FN∆SANDOR1[□]∇
    ∇ N FN∆SANDOR1 PIN;X;LIM;I;J;K;P
[1]   X←Nρ1 ◇ LIM←¯1+2*N ◇ I←1
[2]  L:X[J]←P←((K≠⍳3)/⍳3)[1+2|+/X=K←X[J←(⍬⍴(Nρ2)⍴I)⍳1]]
[3]   'MOVE DISC ';J;' FROM ',PIN[K],' TO ',PIN[P]
[4]   →(LIM≥I←I+1)↑L
    ∇
```

CONTEST NO. 2: TEMPIS FUGIT

REQUIRED: A DYADIC FUNCTION WITH EXPLICIT RESULT OF THE FORM

Z←FROM TIME TO

THAT WORKS AS FOLLOWS:

1)Z←76 1 1 TIME 76 1 28

Z←→27: NUMBER OF DAYS FROM 76 1 1 THROUGH 76 1 28

2)Z←10 TIME 76 1 14

Z←→1976 1 4 1 ; I.E. THE DATE 10 DAYS BEFORE 76 1 14.

NOTE: THE 4TH DIGIT IN THE OUTPUT REPRESENTS THE WEEK DAY
[1=SUNDAY, 7=SATURDAY]

3)Z←1876 12 22 TIME 3

Z←→1876 12 25 2; I.E. THE DATE 3 DAYS AFTER 1876 12 22.

BACKGROUND INFORMATION

1)THE FUNCTION SHOULD WORK FOR ALL DATES BETWEEN OCTOBER 15, 1582,
THE START OF THE GREGORIAN CALENDAR, AND DEC 31, 3999 INCLUSIVE.
THE STATUS OF THE YEAR 4000 IS STILL UNRESOLVED AT PRESS TIME!

2)JANUARY 1, 1600 WAS A SATURDAY.

3)DATE ARGUMENTS SHOULD BE OF THE FORM YY (OR YYYY) MM DD
IF A TWO-DIGIT YEAR IS SPECIFIED, THE CURRENT CENTURY IS ASSUMED
[I.E. 76←→1976]

4)DATE OUTPUT SHOULD BE OF THE FORM YYYY MM DD WW
WHERE WW=1 ←→ SUNDAY, 7 ←→ SATURDAY

ENTRIES WILL BE JUDGED ON CONCISENESS, SPEED, AND ELEGANCE.
PLEASE MAIL/MAILBOX ALL ENTRIES BY DEC 31, 1976.

PLEASE FORWARD ALL COMMENTS, SUGGESTIONS, AND [SIGH] CRITICISMS TO

HAL CARIM [MAILBOX CODE 'HCA']
I.P. SHARP ASSOCIATES,
SUITE 1400, 145 KING STREET WEST,
TORONTO ,ONTARIO,
M5H 1J8,
CANADA.